**Football Match Outcome Prediction using Artificial Neural Network**

**Load required libraries**

First of all, to begin with, the dataset preprocessing and training, the necessary libraries need to be import into the notebook environment. To handle the dataset, here, I mainly used the Pandas library. For plotting various figures, I have used Seaborn and Matplot libraries. Fig. 1 shows the coding example to import all the required libraries for this project.

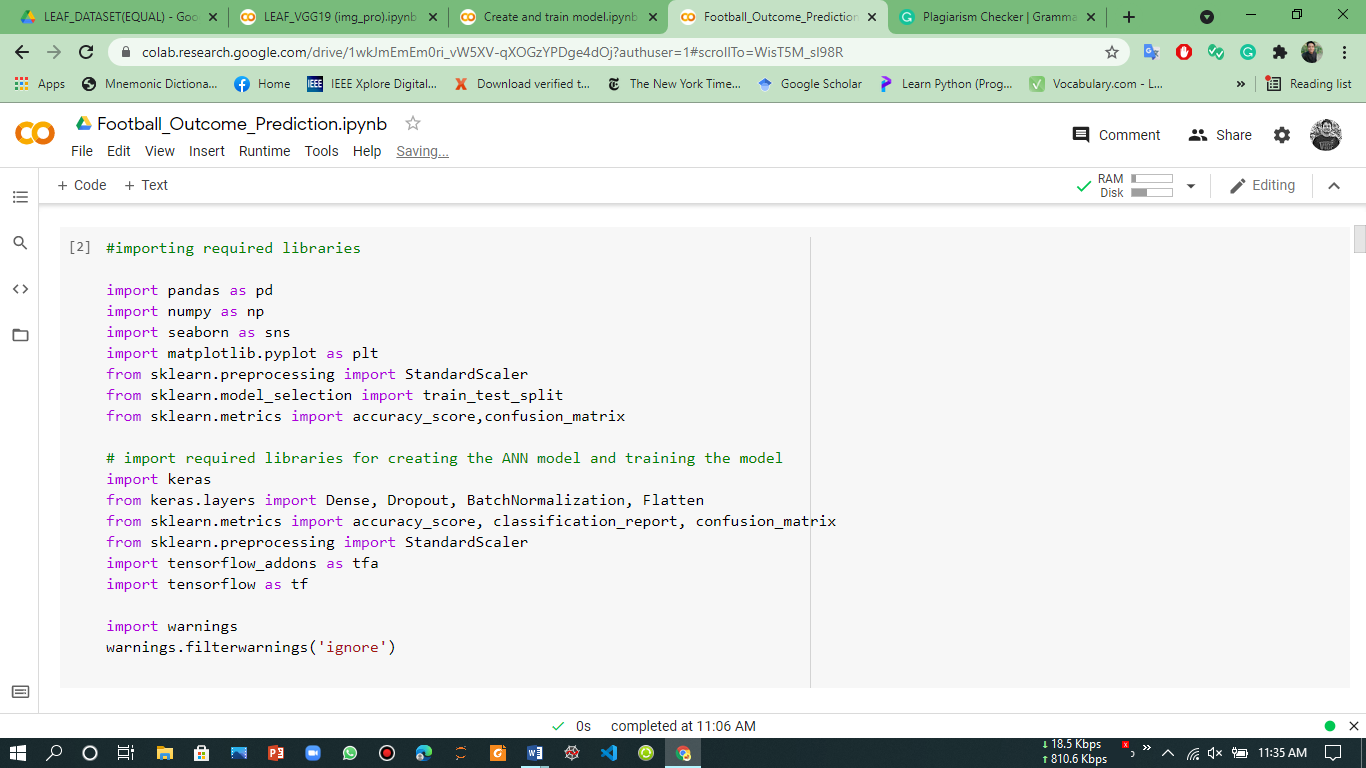


Fig. 1. Code example for importing libraries.

**Dataset Introduction**

The football dataset contains various features to predict the outcome of a specific match. Initially, the dataset has a total of 10 columns or features and 9662 rows. The name of the columns of the dataset are-

* Season
* Date
* Home Team
* Away Team
* HTR - Half Time Result
* FTR - Full Time Result
* HTHG - Half Time Home Goals
* HTAG - Half Time Away Goals
* FTHG - Full Time Home Goals
* FTAG - Full Time Away Goals

**Load Dataset**

To load the dataset, I have used Pandas library which will read the dataset as a CSV format. Fig. 2 shows the code example of loading the dataset using pandas.

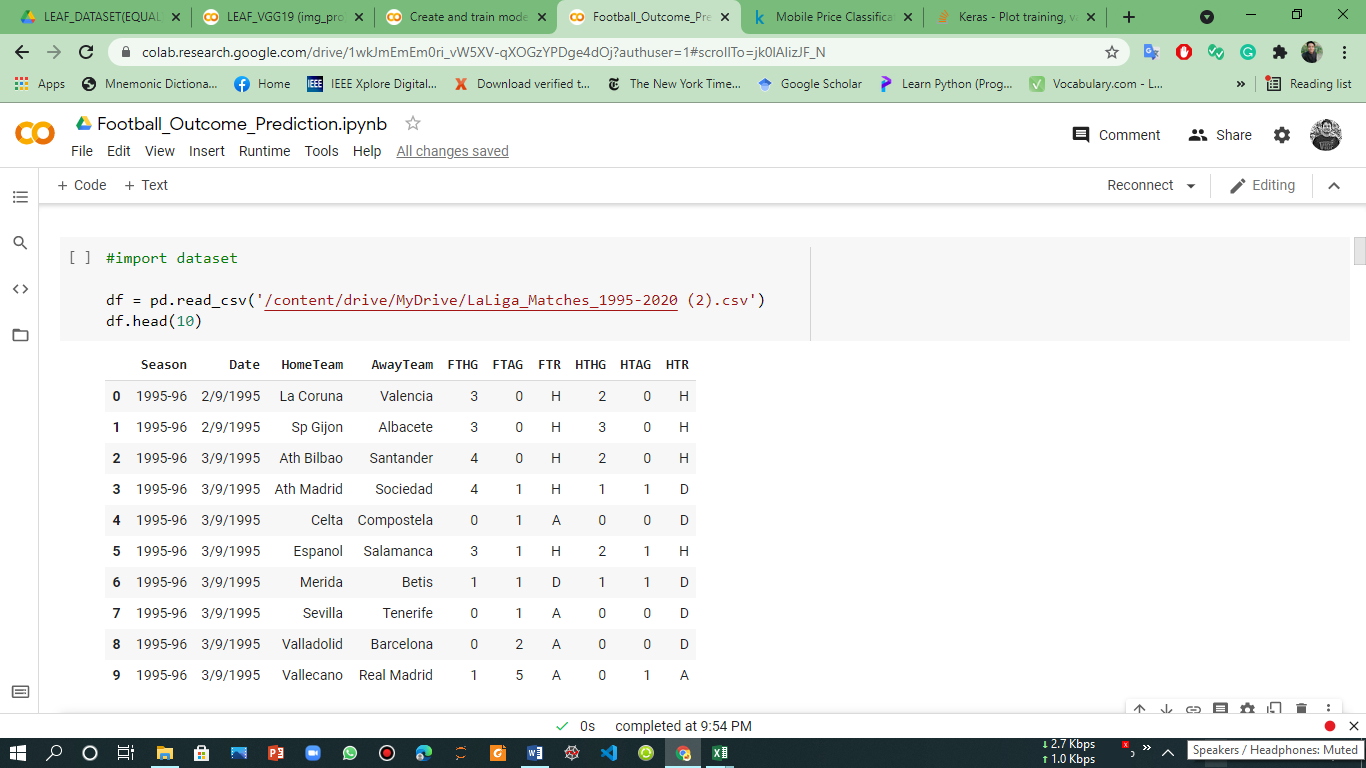


Fig. 2. Loading the dataset using Pandas.

Fig. 2 shows also the first 10 rows of the main dataset where the each columns are representing their corresponding values.

**Cleaning and representation of the dataset**

First, the dataset needs to be clean and finalizing with required and supported data types to begin the work with the dataset. Also, the dataset should not include any empty or null values, which can make our model perform poorly. To analyze the data types of values containing each column, the following code need to be run on the coding environment.

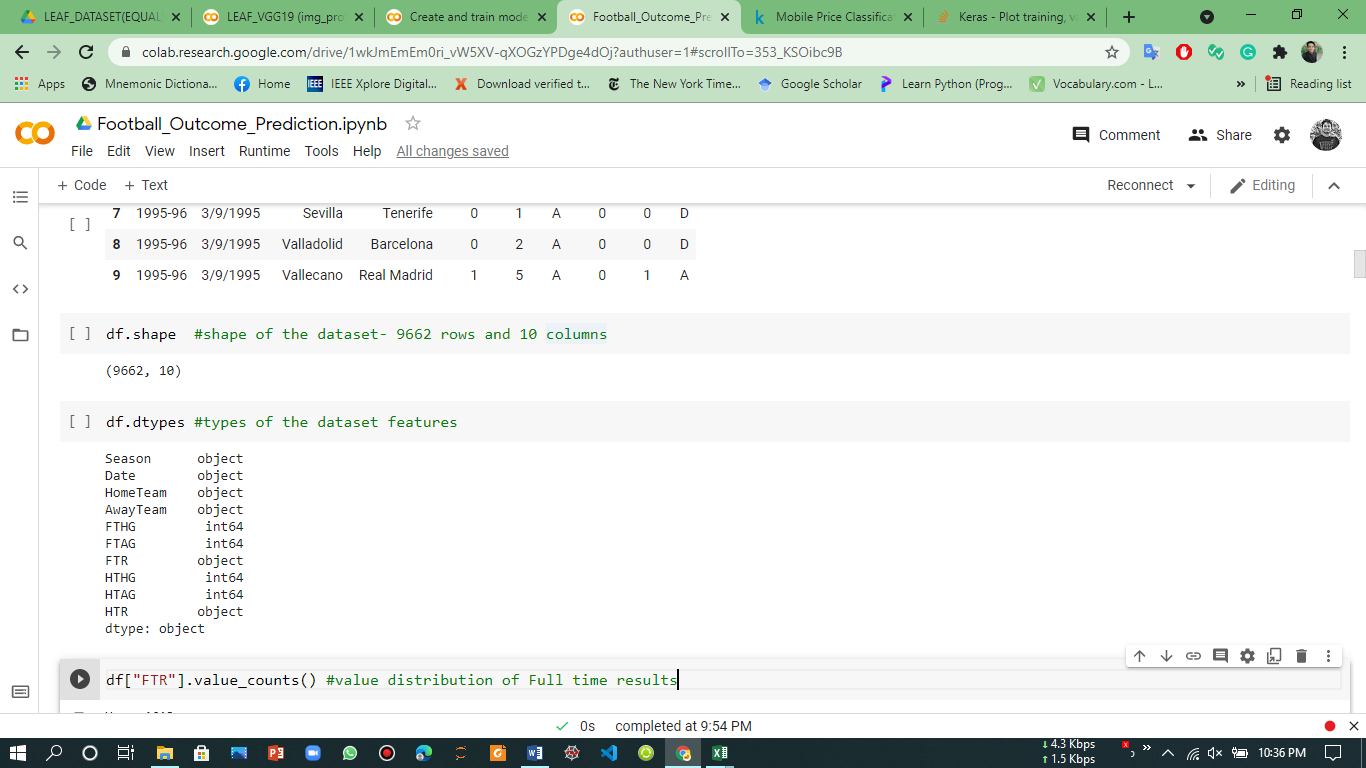


Fig. 3. Coding example of the data types of the values of each column.

Fig. 3 shows that the shape of the dataset is (9662, 10) where 9662 represents the rows and 10 represents the columns. The next segment of the code showing the output of the data type of values of each column. The output shows that among the 10 features six (6) have object data types and four (4) have int64 data types.

For this project, the targeted value/column which needs to be predicted is the FTR (Full Time Result) column. In the CSV file, this column has contains three (3) values- H (Home Team Win), A (Away Team Win), and D (Draw). So, these values are string types. To train the model, this value should be converted to numerical values. To do so, I have employed the find and replacing techniques of pandas which will reform the values of the FTR column into numerical values. Then, those values need to be replaced in the dataset. Similarly, the HTR column contains the same values, so, this column also needs to be converted. Fig. 4 shows the coding examples of the implementation of converting to numerical values.

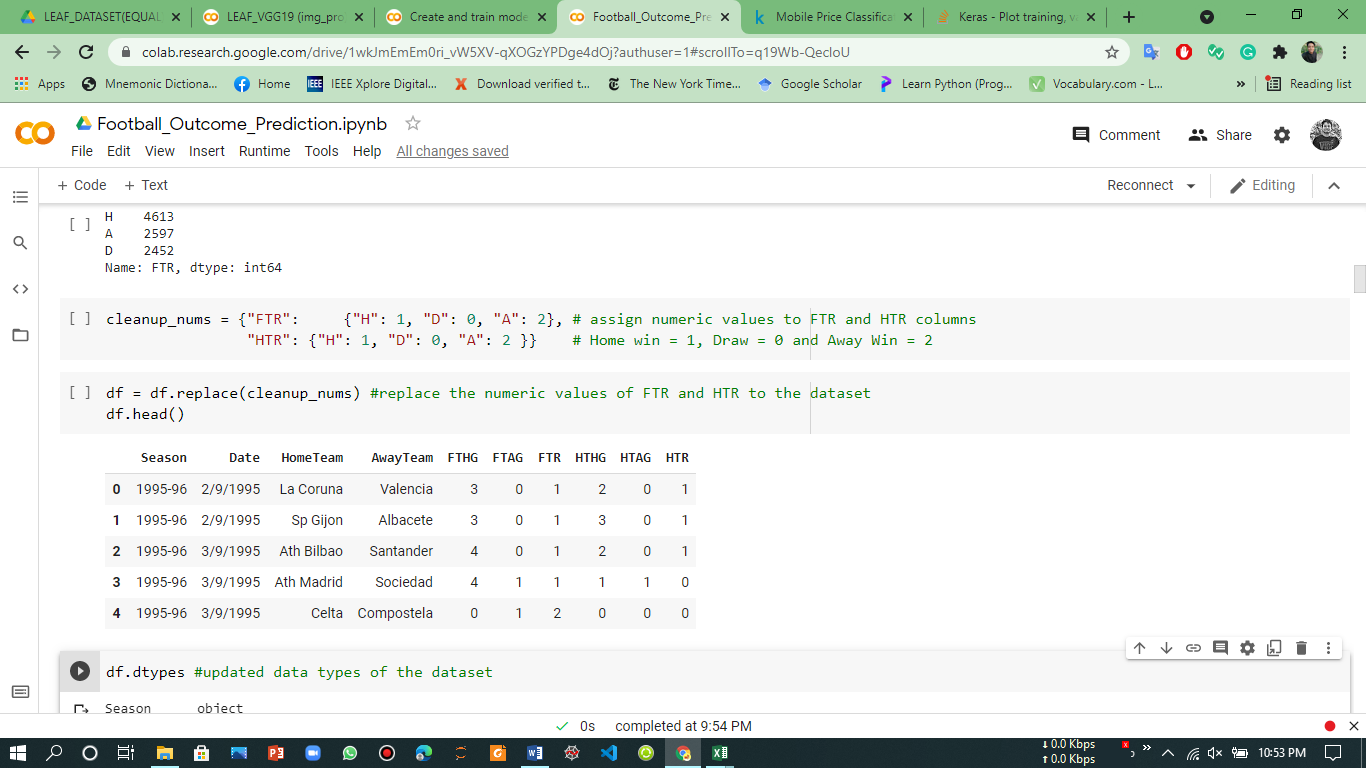


Fig. 4. Coding examples of label encoding of the FTR and HTR columns.

Fig. 4 showing the implementation of the converting and replacing the converted value to the main dataset. The ‘H’ value converted as 1, ‘D’ value converted as 0, and the ‘A’ value converted as 2. The same procedure has been followed in the HTR column also.

From fig. 3, there are also two columns named- HomeTeam and AwayTeam, these two columns are the object data types. These are categorical variables. So, these two columns need to be converted to the numerical value. For this, at first, the values need to be converted as the category data types. Then, the Label Encoder needs to be employed to encode these values. By doing so, each value of these two columns will contains a unique number. As this column contains the team names, so, each team will have different numerical numbers. The fig. 5 and 6 show the coding representation of the label encoder.

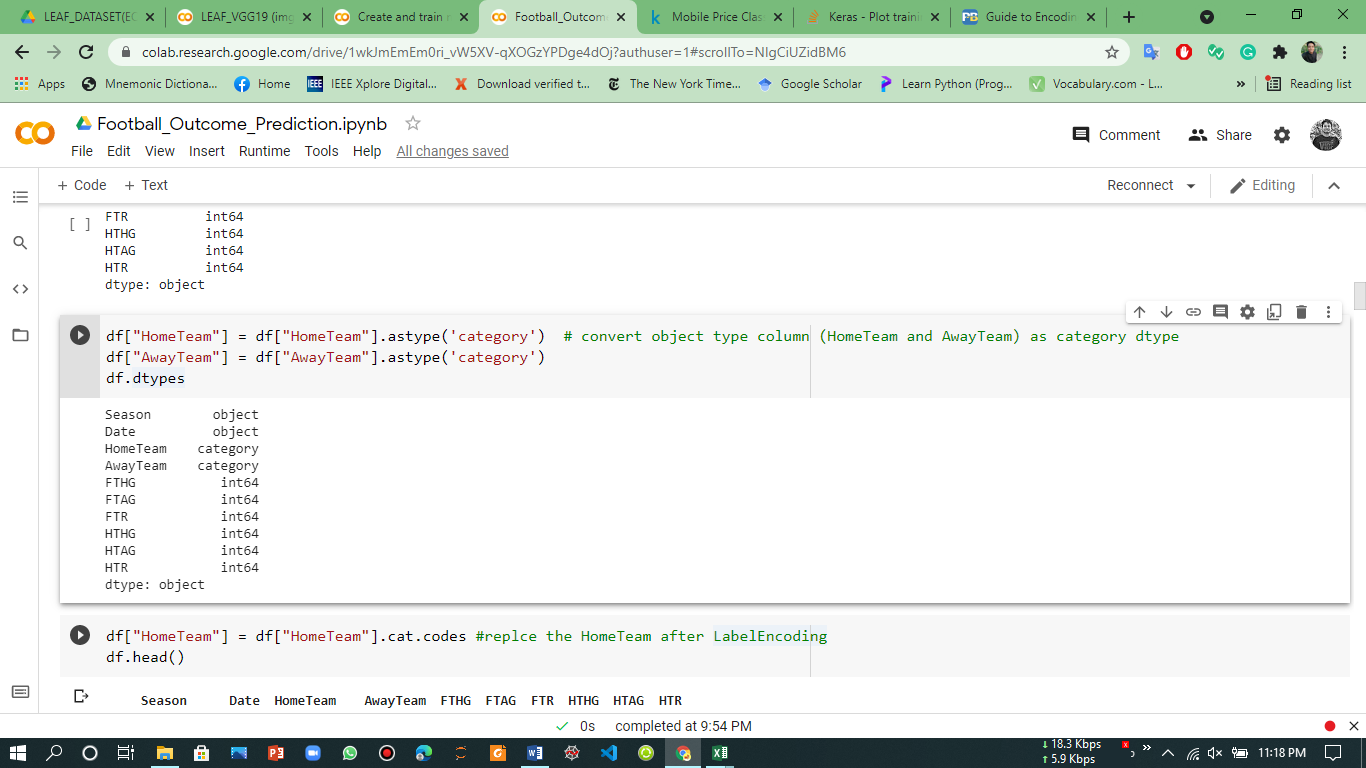


Fig. 5. Coding example of converting the object data type to category data type.

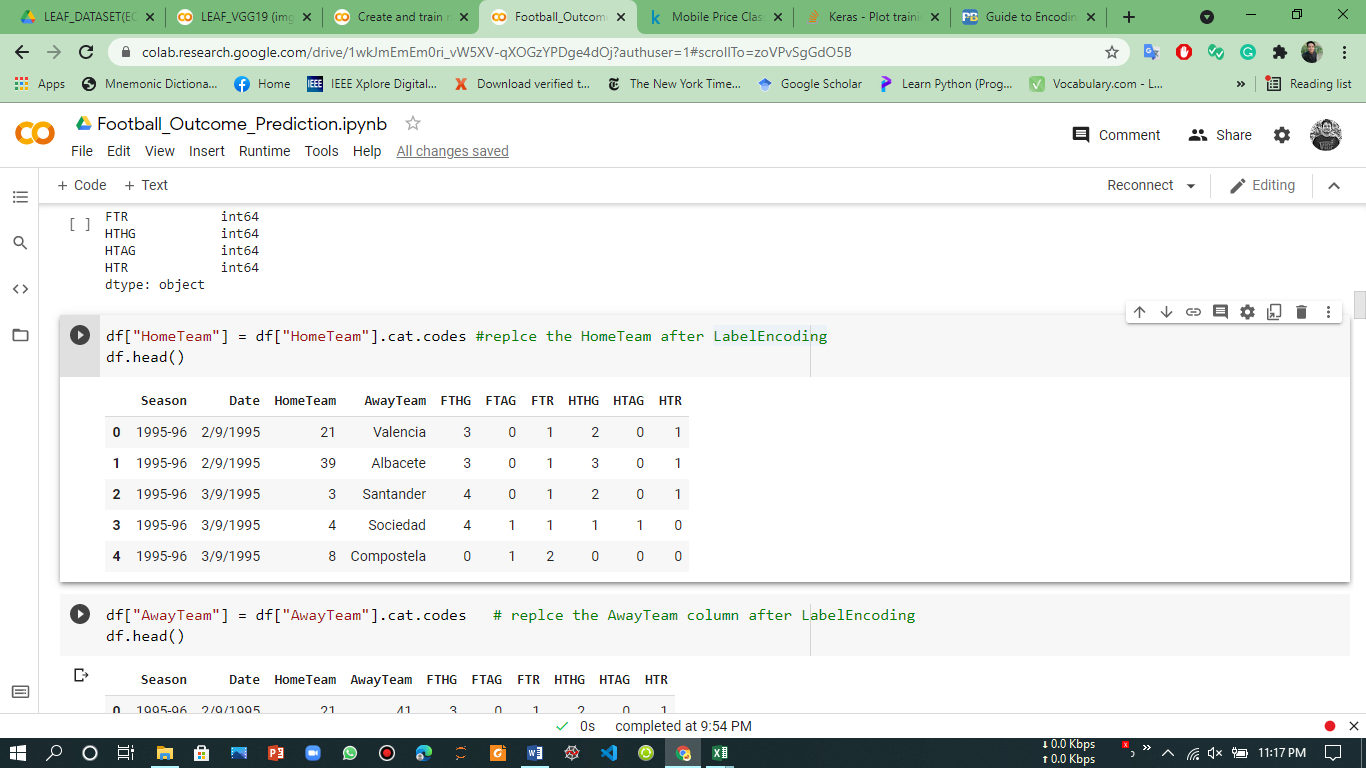


Fig. 6. Coding example of label encoding of the HomeTeam columns

From fig. 6, the values of HomeTeam column is changed from string to integer type and each values are unique, as the team names are unique. Fig. 7 shows the all the unique numbers of the HomeTeam and AwayTeam column.

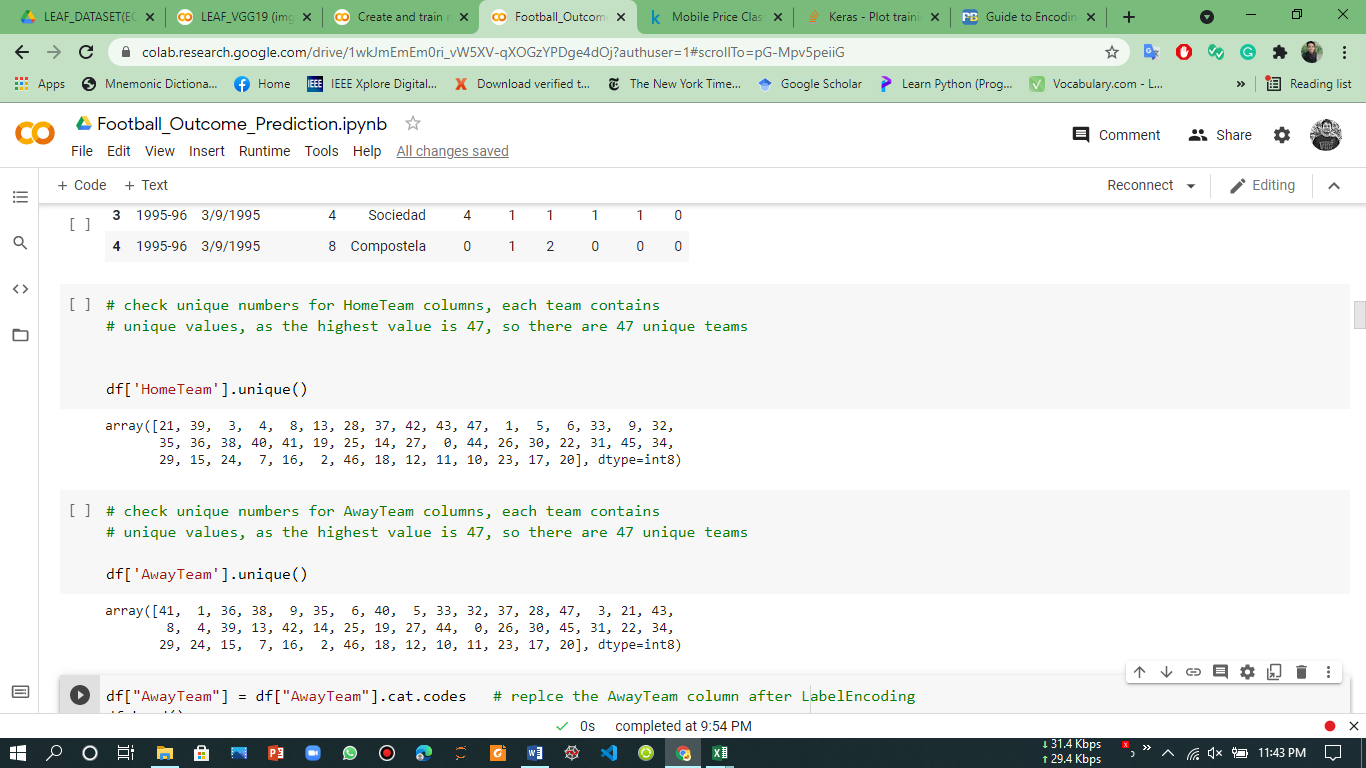


Fig. 7. Output numbers of HomeTeam and AwayTeam Columns.

Now, in the dataset, there are two columns which are not important to predict the match outcome which are Date and Season column. So, these two columns need to be removed from the dataset. To do that, the fig. 8 shows the coding example of removing. Also, the fig. contains the updated and final representation of the dataset.

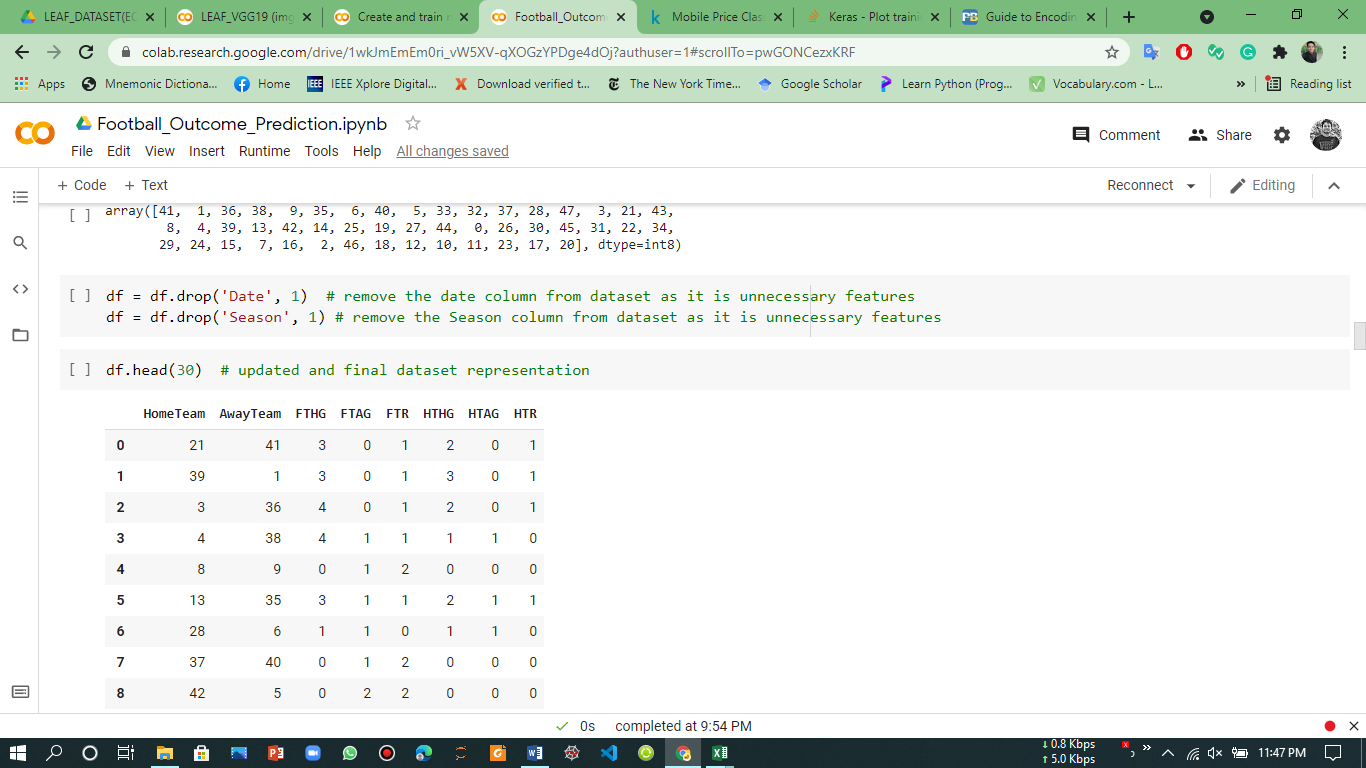


Fig. 8. Removing Date and Season column and representation of updated dataset.

Now, the dataset are fully ready for further operation. Let’s have a look to the updated shape of the dataset and statistics.

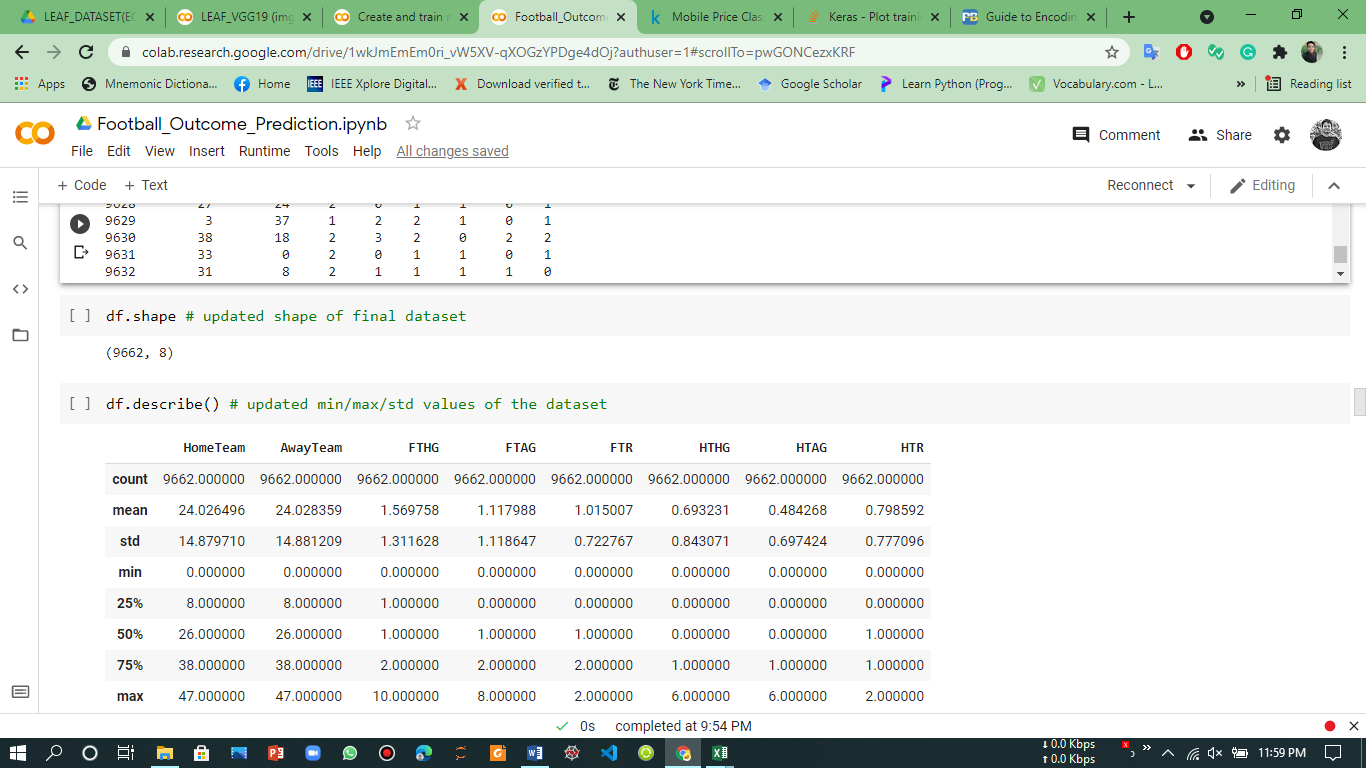


Fig. 9. Coding examples of updated shape and statistics of the final dataset.

Before doing any further approach, let’s check the dataset, if there any null values in any column. Fig. 10 shows the code examples of checking null values.

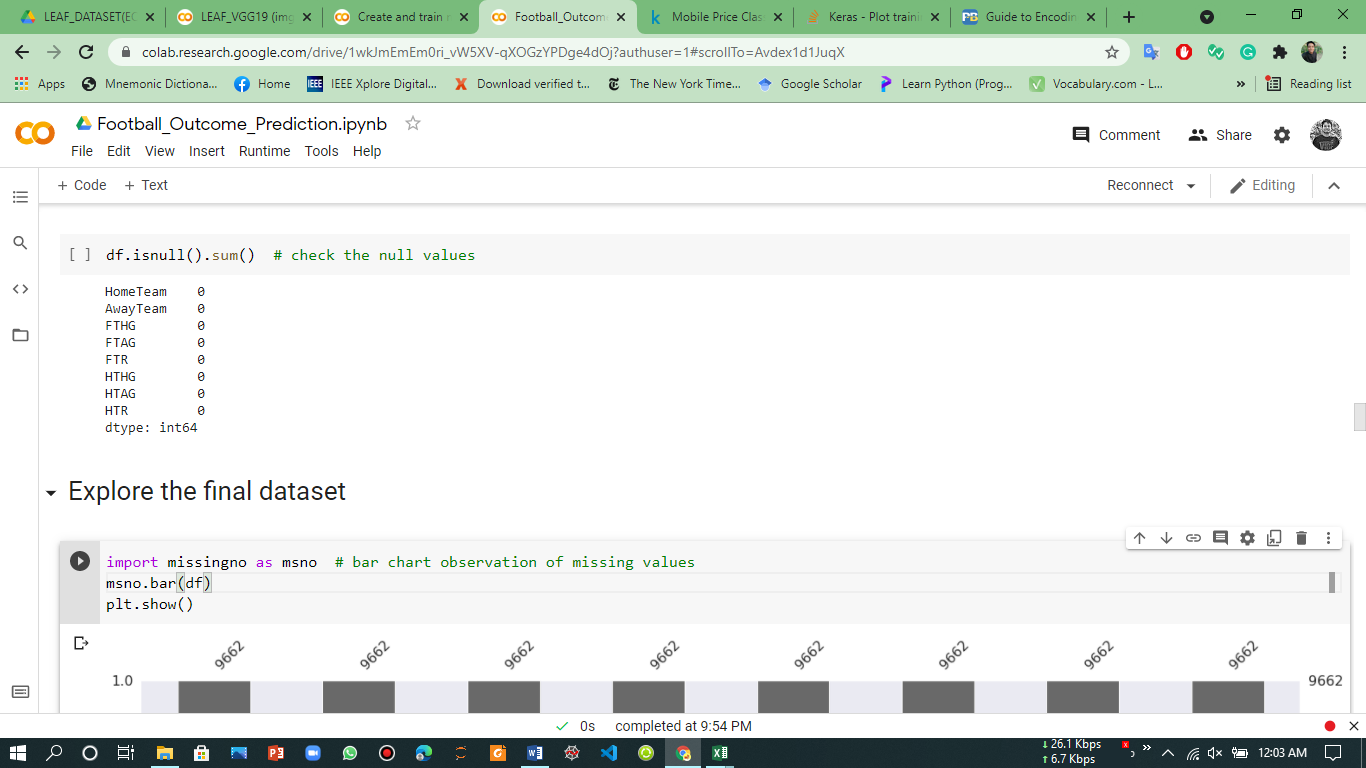


Fig. 10. Coding example for checking null/missing values.

So, from the figure, we can observe that there are no missing values in any column of the dataset. Now, the dataset are ready for training and testing purpose.

**Explore the Dataset**

Before moving to the training, let’s have a look at the target column (FTR) and HTR column by observing the pie chart. Fig. 11 and 12 show the pie chart of the HTR and FTR columns respectively.

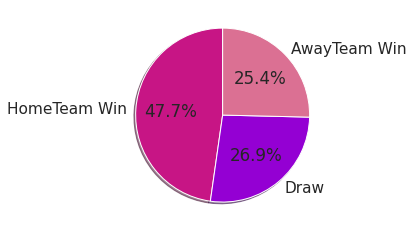
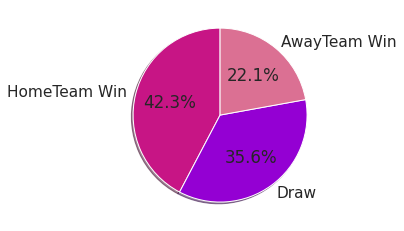


Fig. 11. Pie chart of the HTR column. Fig. 12. Pie chart of FTR column.

From these figures, in the HTR column, the winning percentage of the Home team is 42.3% and in the FTR column is 47.7%. Where the winning percentage of the HTR and FTR column of the Away team is 35.6% and 26.9% respectively. So, the winning percentage of the Home team is much higher than the Away team. Also, to mention the draw percentage of these two columns is 22.1% in HTR and 25.4% in FTR columns.

**Spilt the Dataset**

Before training the model, the dataset needs to be split into train and test sets. Here, I have split the dataset by keeping 80% of the data for training and 20% data for testing purposes. Fig. 13 shows the code implementation of the train and test set to split.

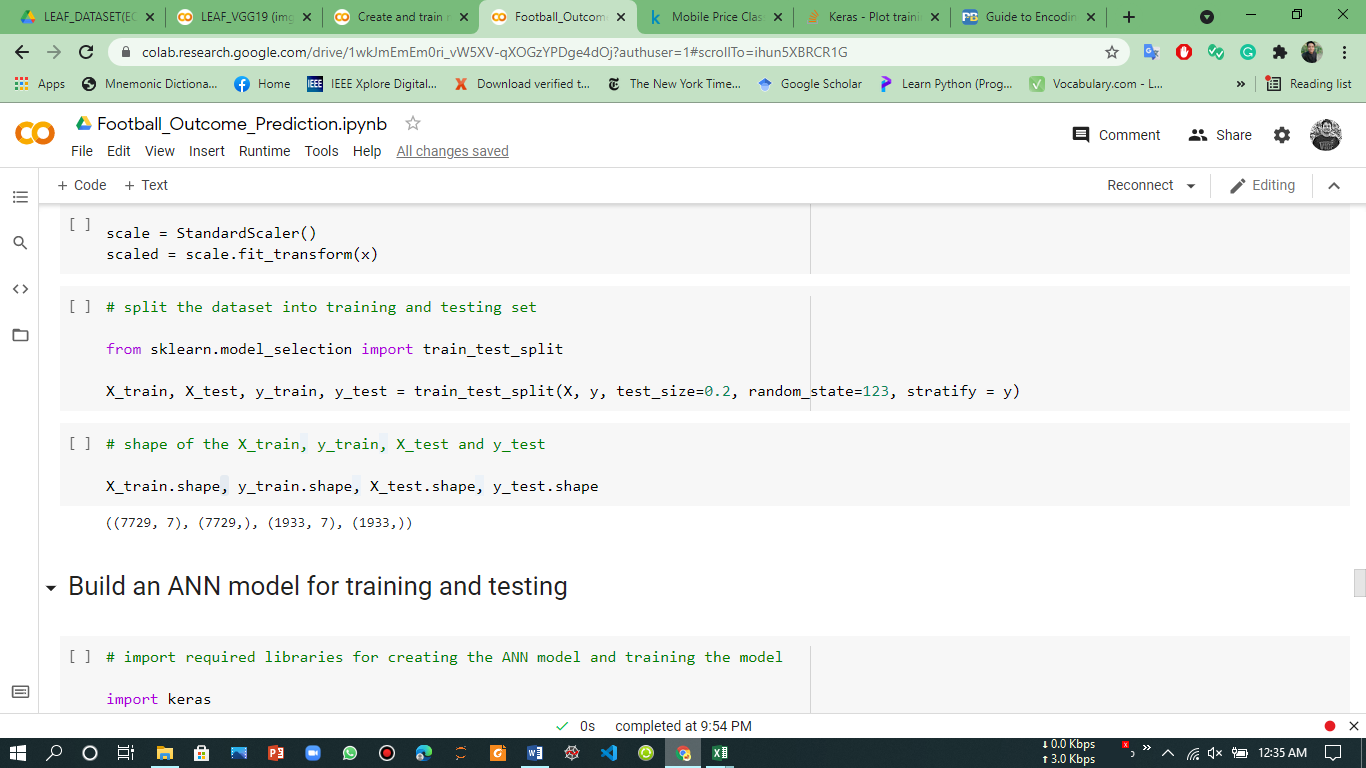


Fig. 13. Coding examples of train-test spilt of the dataset.

From the following figure, after split the dataset, the X\_train and y\_train contain a total of 7729 data, and X\_test and y\_test contain a total of 1933 data from the dataset.

**Build an ANN model**

To train the model, at first, I have built an ANN model from scratch. The model will be used for the training and testing of the dataset. Here, I have used one (1) input layer, three (3) hidden layers, and one (1) output layer. The detailed representation of the model is shown in fig. 14.

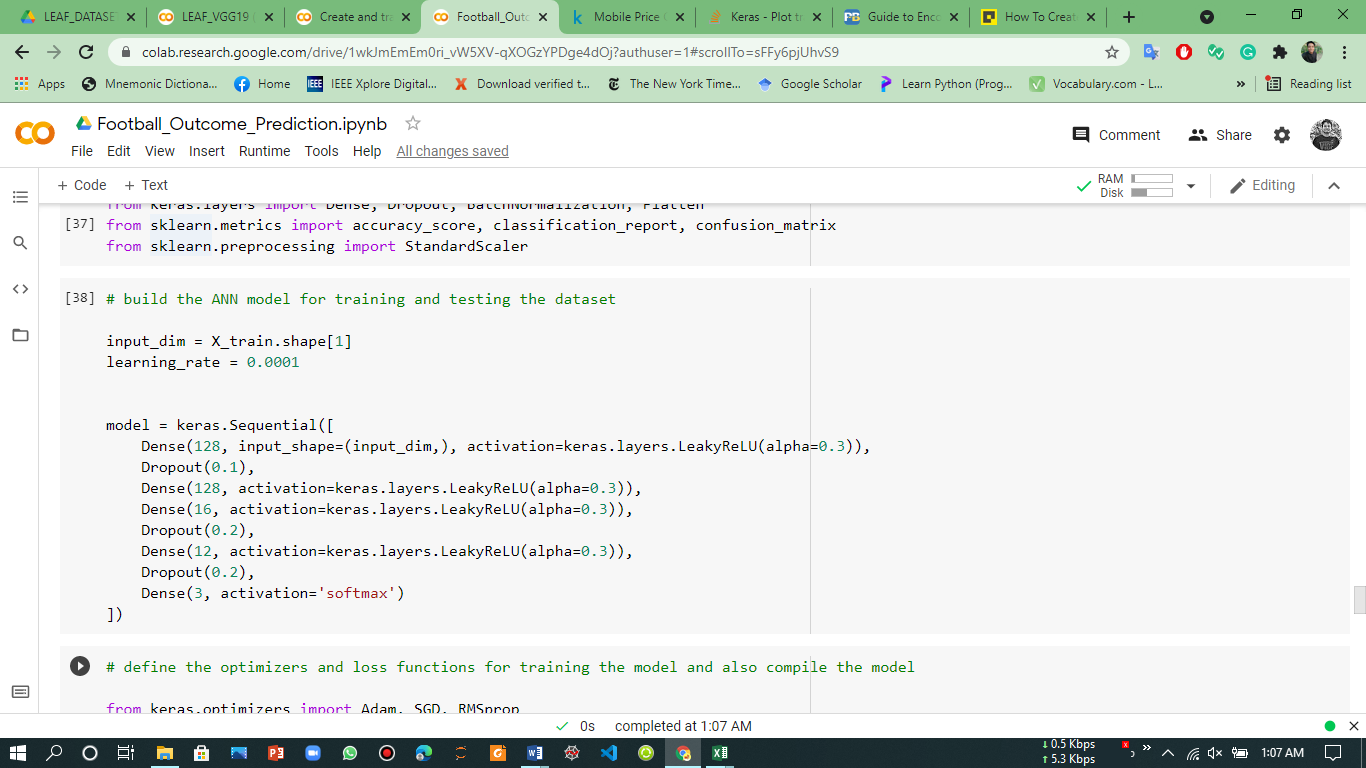


Fig. 14. Representation of the ANN model.

The figure shows that the input layer has nodes of 128 and the activation function is LeakyReLU. After the input layer, there are three dense layers or hidden layers with different nodes but the same activation function (LeakyReLU). Also, there is one output layer with 3 nodes for 3 output classes (HomeTeam win, Draw, or AwayTeam win). The softmax activation function is used in the output layer as this is a multiclass classification problem. To reduce the over-fitting and underfitting issues, the DropOut function is also used. Thus, the model is ready for training and testing.

**Compile and Fit the Model**

After building the ANN model, the model needs to be compiled by employing optimizers and loss function. Here, I have used Adam optimizer and for calculating the loss, the sparse categorical cross-entropy is used. For training the model, I have set the batch size to 64 and a learning rate of 0.0001 with only 20 epochs. Fig. 15 shows the detailed view of compile and fitting the model.

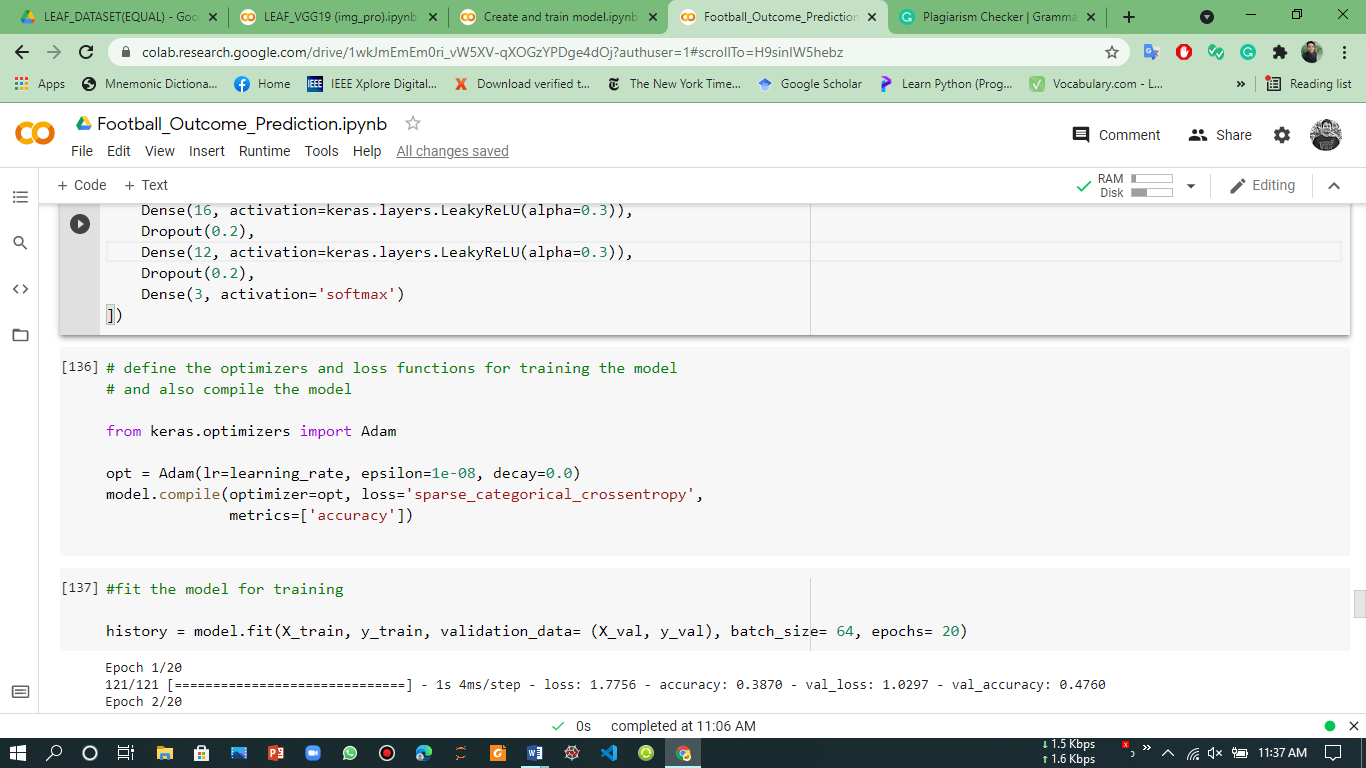


Fig. 15. Coding examples of compile and fitting the ANN model.

**Performance Analysis**

To analyze the performance of the training and testing, I have considered accuracy, precision, recall, and F1 score. After training the model with 64 batch sizes and 20 epochs, the model has been achieved a satisfactory result. The result of these four (4) metrics has been shown in Table I.

Table I. Analysis of various metrics sore

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameters/ Metrics | **Accuracy** | **Recall** | **Precision** | **F1 Score** | **Test Accuracy** |
| Draw | 98.93% | 98% | 100% | 99% | 99.38% |
| Home Team | 100% | 99% | 100% |
| Away Team | 100% | 99% | 100% |

The performance table shows that the overall training accuracy is 99.93%. Where the overall test accuracy is 99.38%. These two scores indicating the good performance of the ANN model to predict or classify the football dataset. Also, the recall, precision, and f1 score performances are very satisfactory in each class. In draw class, the precision achieved 100% accuracy, whereas in Home and Away class, recall and f1 score achieved 100% accuracy. So, the model has been performed well to predict the football outcomes.

**Confusion Matrix**

The confusion matrix is an important method to measure the prediction accuracy of the ANN model. It plots actual or true labels concerning the predicted label for each class. Fig. 16 shows the detailed view of the confusion matrix of this ANN model.

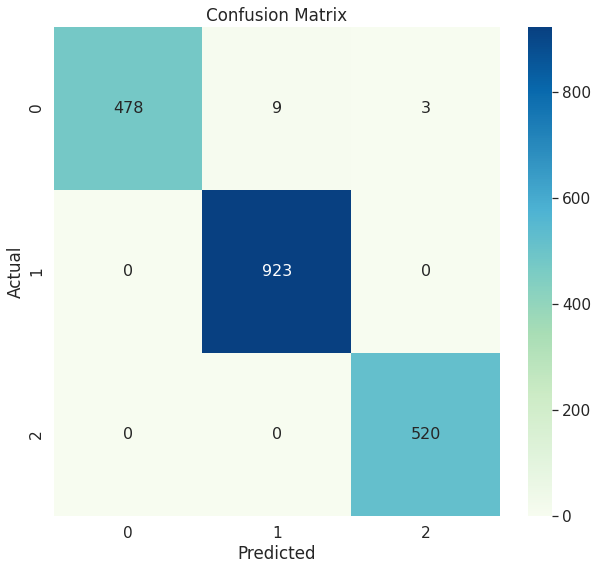


Fig. 16. Confusion matrix.

The confusion matrix shows the prediction of each three class with respect to the actual label. There are three classes where ‘0’ represents the ‘Draw’, ‘1’ represents the ‘HomeTeam Win’, and ‘2’ represents the ‘Away Team Win’. In actual label, draw class has 490 values and the ANN model predicted 478 values, so, it miss-predict the 12 values as other class. Whereas in the other classes (Home and Away team win), the model successfully predicted all the values as a true label. In other words, the model hasn’t done any miss-prediction to predict the Home and Away team win classes. Thus, it can be said that the ANN model has shown a very satisfactory performance to predict the outcomes of the given football match dataset.